

Software

IDC DOCUMENTATION

Subscription Subsystem



**Approved for public release;
distribution unlimited**

Notice

Every effort was made to ensure that the information in this document was accurate at the time of printing. However, the information is subject to change.

Contributors

David Salzberg, Science Applications International Corporation

Trademarks

Ethernet is a registered trademark of Xerox Corporation.

ORACLE is a registered trademark of Oracle Corporation.

Solaris is a registered trademark of Sun Microsystems.

SPARC is a registered trademark of Sun Microsystems.

SQL*Plus is a registered trademark of Oracle Corporation.

UNIX is a registered trademark of UNIX System Labs, Inc.

Ordering Information

This document was issued by the Geophysical Systems Operation of Science Applications International Corporation (SAIC) as part of the International Data Centre (IDC) Documentation. The ordering number for this document is SAIC-98/3001, published April 1998. Copies of this document may be ordered by FAX: (619) 458-4993.

This document is cited within other IDC documents as [IDC7.4.4].

Subscription Subsystem

CONTENTS

About this Document	i
■ PURPOSE	ii
■ SCOPE	ii
■ AUDIENCE	iii
■ RELATED INFORMATION	iii
■ USING THIS DOCUMENT	iv
Conventions	v
Overview	1
■ INTRODUCTION	2
■ FUNCTIONALITY	3
■ IDENTIFICATION	6
■ STATUS OF DEVELOPMENT	6
■ BACKGROUND AND HISTORY	7
■ OPERATING ENVIRONMENT	8
Hardware	8
Commercial-Off-the-Shelf Software	8
Architectural Design	9
■ CONCEPTUAL DESIGN	10
■ DESIGN DECISIONS	16
Programming Language	16
Global Libraries	16
Database	16
Interprocess Communication (IPC)	17
File System	17

UNIX Mail	17
FTP	17
Web	17
Design Model	18
Database Schema Overview	18
■ FUNCTIONAL DESCRIPTION	20
Receiving Subscription Requests	22
Registering Subscriptions	22
Processing and Formatting Subscriptions	23
Error Handling	23
Tracking Subscriptions	24
■ INTERFACE DESIGN	24
Interface with Other IDC Systems	24
Interface with External Users	25
Interface with Operators	25
Detailed Design	27
■ DATA FLOW MODEL	28
■ SOFTWARE UNITS	29
DataReady_to_Prodtrack	30
FlatProduct_to_DataReady	31
Fpstacap	31
ParseSubs	31
SubsProcess	33
SubsTrack	35
SubsWatch	35
Write_fp	35
■ DATABASE DESCRIPTION	35
Database Design	35
Database Schema	36
Requirements	49
■ INTRODUCTION	50
■ GENERAL REQUIREMENTS	50

■ FUNCTIONAL REQUIREMENTS	51
Storage of User Profiles and Subscription Records	51
Subscription Configuration	51
Subscription Maintenance	52
Subscription Distribution	52
Logging/Monitoring	53
■ SYSTEM REQUIREMENTS	54
■ REQUIREMENTS TRACEABILITY	55
References	63
Glossary	G1

Subscription Subsystem

FIGURES

FIGURE 1.	IDC SOFTWARE CONFIGURATION HIERARCHY	4
FIGURE 2.	RELATIONSHIP OF SUBSCRIPTION SUBSYSTEM TO OTHER SOFTWARE UNITS OF DATA SERVICES CSCI	5
FIGURE 3.	REPRESENTATIVE HARDWARE CONFIGURATION FOR SUBSCRIPTION SUBSYSTEM	8
FIGURE 4.	OBJECT MODEL OF SUBSCRIPTION SUBSYSTEM	11
FIGURE 5.	PROCESSING FLOW AND CREATION OF DATA PRODUCTS	14
FIGURE 6.	FLOW OF SUBSCRIPTION PARSING	20
FIGURE 7.	DATA FLOW AND EXTERNAL INTERFACES OF SUBSCRIPTION SUBSYSTEM	21
FIGURE 8.	ENTITY-RELATIONSHIP MODEL OF SUBSCRIPTION SUBSYSTEM TABLES	37

Subscription Subsystem

TABLES

TABLE I:	DATA FLOW SYMBOLS	v
TABLE II:	ENTITY-RELATIONSHIP SYMBOLS	vi
TABLE III:	MISCELLANEOUS SYMBOLS	vi
TABLE IV:	TYPOGRAPHICAL CONVENTIONS	vii
TABLE V:	TERMINOLOGY	viii
TABLE 1:	COMPONENTS OF SUBSCRIPTION SUBSYSTEM	6
TABLE 2:	STANDARD PRODUCTS AVAILABLE THROUGH SUBSCRIPTION SUBSYSTEM	12
TABLE 3:	TERMS RELATING TO STANDARD PRODUCTS	15
TABLE 4:	DATABASE TABLES USED BY SUBSCRIPTION SUBSYSTEM	19
TABLE 5:	DATAREADY	38
TABLE 6:	FPDESCRIPTION	39
TABLE 7:	FILEPRODUCT	40
TABLE 8:	PRODTRACK	41
TABLE 9:	PRODUCTCRITERIA	42
TABLE 10:	PRODUCTTYPE(*)	43
TABLE 11:	PRODUCTTYPEORIGIN	44
TABLE 12:	PRODUCTYPESTA	45
TABLE 13:	SUBS	46
TABLE 14:	SUBSUSER	47
TABLE 15:	TRACEABILITY OF GENERAL REQUIREMENTS	55
TABLE 16:	TRACEABILITY OF FUNCTIONAL REQUIREMENTS: STORAGE OF USER PROFILES AND SUBSCRIPTION RECORDS	56
TABLE 17:	TRACEABILITY OF FUNCTIONAL REQUIREMENTS: SUBSCRIPTION CONFIGURATION	57
TABLE 18:	TRACEABILITY OF FUNCTIONAL REQUIREMENTS: SUBSCRIPTION MAINTENANCE	58

TABLE 19:	TRACEABILITY OF FUNCTIONAL REQUIREMENTS: SUBSCRIPTION DISTRIBUTION	59
TABLE 20:	TRACEABILITY OF FUNCTIONAL REQUIREMENTS: LOGGING/MONITORING	61

About this Document

This chapter describes the organization and content of the document and includes the following topics:

- Purpose
- Scope
- Audience
- Related Information
- Using This Document

About this Document

PURPOSE

This document describes the design and requirements of the Subscription Subsystem software of the International Data Centre (IDC). The software is a computer software component (CSC) of the Data Services Computer Software Configuration Item (CSCI). This document provides a basis for implementing, supporting, and testing the software.

This document is Issue 1 of an expected sequence of increasingly refined descriptions of the Subscription Subsystem software. It supersedes the design and requirement descriptions contained in previous informal memoranda and Configuration Control Board proposals.

SCOPE

The Subscription Subsystem software is identified as follows:

Title:	Subscription Subsystem
Abbreviation:	(none)
Identification Number:	CSC 4.4
Version Number:	1

This document describes the architectural and detailed design of the software including its functionality, components, data structures, high-level interfaces, method of execution, and underlying hardware. Additionally, this document specifies the requirements of the software and its components. The information contained in this document is modeled on the Data Item Description for Software Design Descriptions [DOD94a] and Software Requirements Specification [DOD94b].

AUDIENCE

This document is intended for all engineering and management staff concerned with the design and requirements of all IDC software in general and of the Subscription Subsystem in particular. The detailed descriptions are intended for programmers who will be developing, testing, or maintaining the Subscription Subsystem.

RELATED INFORMATION

The following UNIX Manual Pages apply to the Subscription Subsystem software:

- SubProcess
- ParseSubs

The following documents complement this document:

- *IDC Database Schema* [Car97]
- *Adding Station Capability and Flat Products to the Subscription Subsystem* [Sal97]

See "References" on page 63 for a list of documents that supplement this document.

USING THIS DOCUMENT

The documentation of the IDC software is grouped within Category 7 of the overall documentation architecture, as charted on the Roadmap located on the pages preceding the Table of Contents. Within Category 7, the documentation is further subdivided into the same six CSCIs as the software architecture. The highlighted box represents this document, the Subscription Subsystem Software.

This document is organized as follows:

- Overview
This chapter provides a high-level view of the Subscription Subsystem, including its functionality, components, background, status of development, and current operating environment.
- Architectural Design
This chapter describes the architectural design of the Subscription Subsystem, including its conceptual design, design decisions, functions, and interface design.
- Detailed Design
This chapter describes the detailed design of the Subscription Subsystem including its data flow, software units, and database design.
- Requirements
This chapter describes the general, functional, and system requirements of the Subscription Subsystem. Traceability tables define how the general and functional requirements are met.
- References
This section lists the sources cited in this document.
- Glossary
This section defines the terms, abbreviations, and acronyms used in this document.

Conventions

This document uses a variety of conventions, which are described in the following tables. Table I shows the conventions (Gane-Sarson) for data flow diagrams. Table II shows the conventions for entity-relationship diagrams. Table III shows symbols used to indicate levels of the documentation hierarchy. Table IV illustrates typographical conventions. Table V explains certain technical terms that are not part of the standard Glossary, which is found at the end of this document.

TABLE I: DATA FLOW SYMBOLS

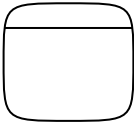
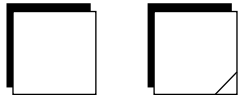

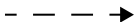

Description	Symbol
process	
external source or sink of data (left) duplicated external source or sink of data (right)	
data store (left) duplicated data store (right)	
control flow	
data flow	

TABLE II: ENTITY-RELATIONSHIP SYMBOLS







Description	Symbol
One A maps to one B.	A  B
One A maps to zero or one B.	A  B
One A maps to many Bs.	A  B
One A maps to zero or many Bs.	A  B
database table	<div> <div>tablename</div> <div>  <i>primary key</i>  <i>foreign key</i> </div> <div> attribute 1 attribute 2 . . . attribute n </div> </div>

TABLE III: MISCELLANEOUS SYMBOLS


Description	Symbol
category of documentation	
subcategory of documentation	
document	

TABLE IV: TYPOGRAPHICAL CONVENTIONS

Element	Font	Example
database table	bold	dataready
database table and attribute, when written in the dot notation		prodtrack.status
database attributes	<i>italics</i>	<i>status</i>
processes, software units, and libraries		<i>ParseSubs</i>
user-defined arguments and variables used in parameter (par) files or program command lines		<i>delete-remarks object</i>
titles of documents		<i>Subscription Subsystem Software User Manual</i>
computer code and output	courier	>(list 'a 'b 'c)
filenames, directories, and websites		amr.prn
text that should be typed in exactly as shown		edit-filter-dialog

TABLE V: TERMINOLOGY

Term	Description
fork	This UNIX system routine is used by a parent process to create a child process.
instance	An instance describes a running computer program. An individual program may have multiple instances on one or more host computers.
object model	An object model describes the structure and behavior of objects.
product	A product is a standard collection of data.
software unit or computer software component	These terms define an element of a Computer Software Configuration Item (CSCI) including a major subdivision of a CSCI or any of its contained subunits.
subscriber	A subscriber is an entity (usually a person), which has placed a standing order to receive IDC products.

Overview

This chapter provides a general overview of the Subscription Subsystem software and includes the following topics:

- Introduction
- Functionality
- Identification
- Status of Development
- Background and History
- Operating Environment

Overview

INTRODUCTION

The software of the IDC acquires timeseries and radionuclide data from stations of the International Monitoring System (IMS) and other locations. These data are passed through a number of automatic and interactive analysis stages, which culminate in the estimation of location and in the origin time of events (earthquakes, volcanic eruptions, and so on) in the earth, including its oceans and atmosphere. The results of the analysis are distributed to States Parties and other users by various means. Approximately one million lines of developmental software are spread across seven computer software configuration items (CSCIs) of the software architecture. Two additional CSCIs are devoted to non-developmental software and run-time data of the software. Figure 1 shows the logical organization of the IDC software. The Data Services CSCI receives, archives, and distributes data through the following computer software components (CSCs):

- Continuous Data Subsystem

This software acquires timeseries data according to a standard protocol and forwards the data to external users [IDC3.4.2].

- Message Subsystem

This software exchanges data in response to user requests. The data are formatted according to a standard protocol and exchanged through UNIX mail [IDC3.4.1]. This software also provides the interface to mail for the Retrieve and Subscription Subsystems.

- Retrieve Subsystem

This software prepares messages, formatted according to the standard protocol, that retrieve segments of data from stations of the IMS auxiliary seismic network [IDC3.4.1]. The software also parses the response messages. The Message Subsystem exchanges the messages.

- Subscription Subsystem

This software maintains a subscriber database and prepares the regular data products for delivery to subscribers. The Message Subsystem receives the subscription requests and delivers the subscription products.

- Data Archive Subsystem

This software saves timeseries data to near-line and off-line archives and recovers data from the archives.

- Website Subsystem

This software runs the IDC Web site.

The relationship of the Subscription Subsystem to other components of the Data Services CSCI is indicated in Figure 2. This figure shows how product subscribers (g) submit requests via the Message Subsystem and subscription data are stored in the database (D1). Furthermore, subscribers' data, when available, are queued for orderly formatting and delivery to the subscriber, again via the Message Subsystem.

FUNCTIONALITY

The Subscription Subsystem enables authorized users to place orders for standard and customized IDC products and to have the products delivered on a routine basis, or alternatively, as quickly as they become available. When data are available for distribution, the Subscription Subsystem determines which orders need the data. The Subscription Subsystem then constructs order descriptions and passes the descriptions in the form of GSE Request Messages to the Message Subsystem (see "Formats and Protocols" in [GSE95b]). The Message Subsystem completes the order and mails the products (or product announcement if the file is too large) to subscribers. The Subscription Subsystem tracks delivery of the product; this auditing supports recovery after system failures.

Overview ▼

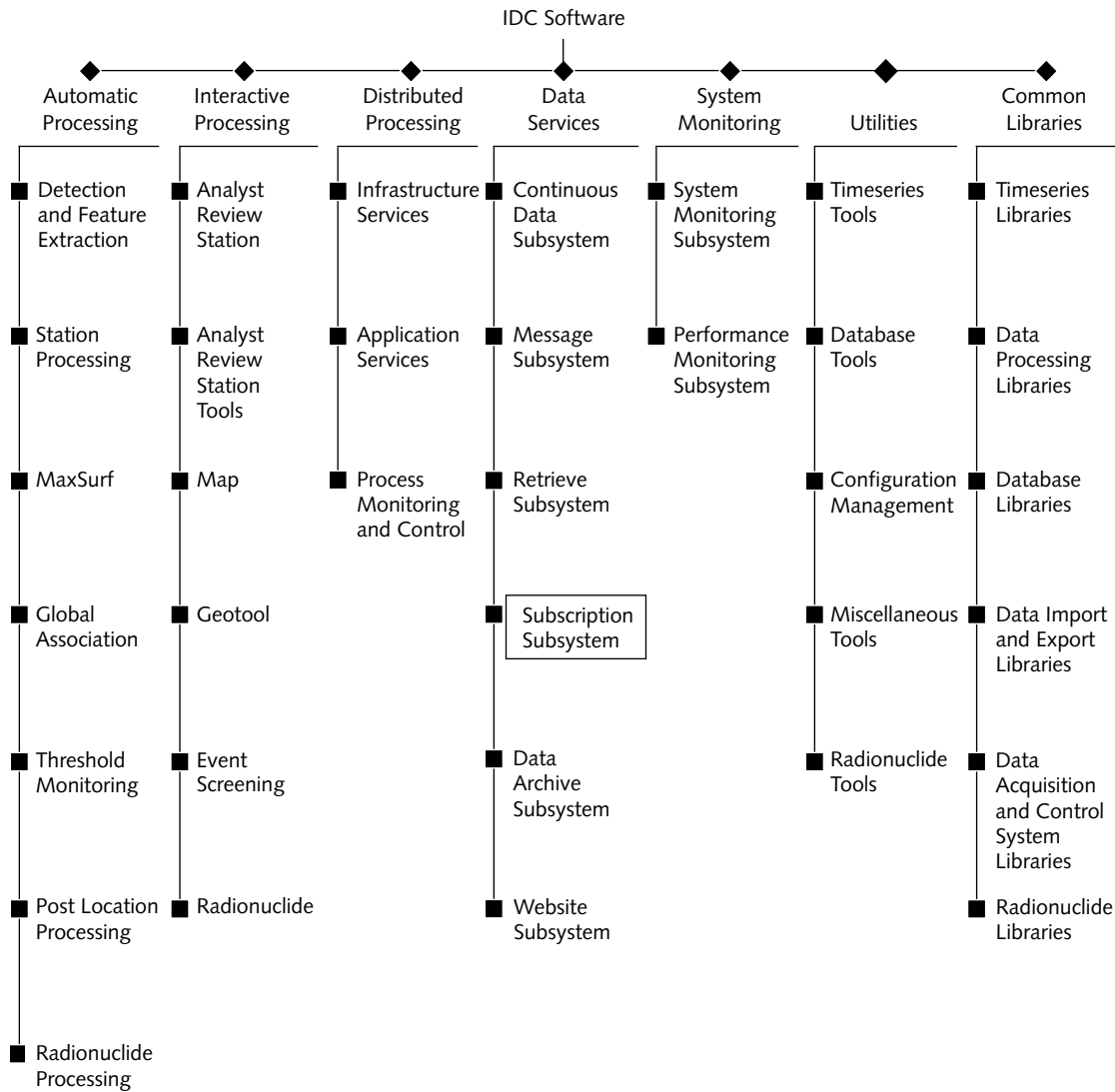


FIGURE 1. IDC SOFTWARE CONFIGURATION HIERARCHY

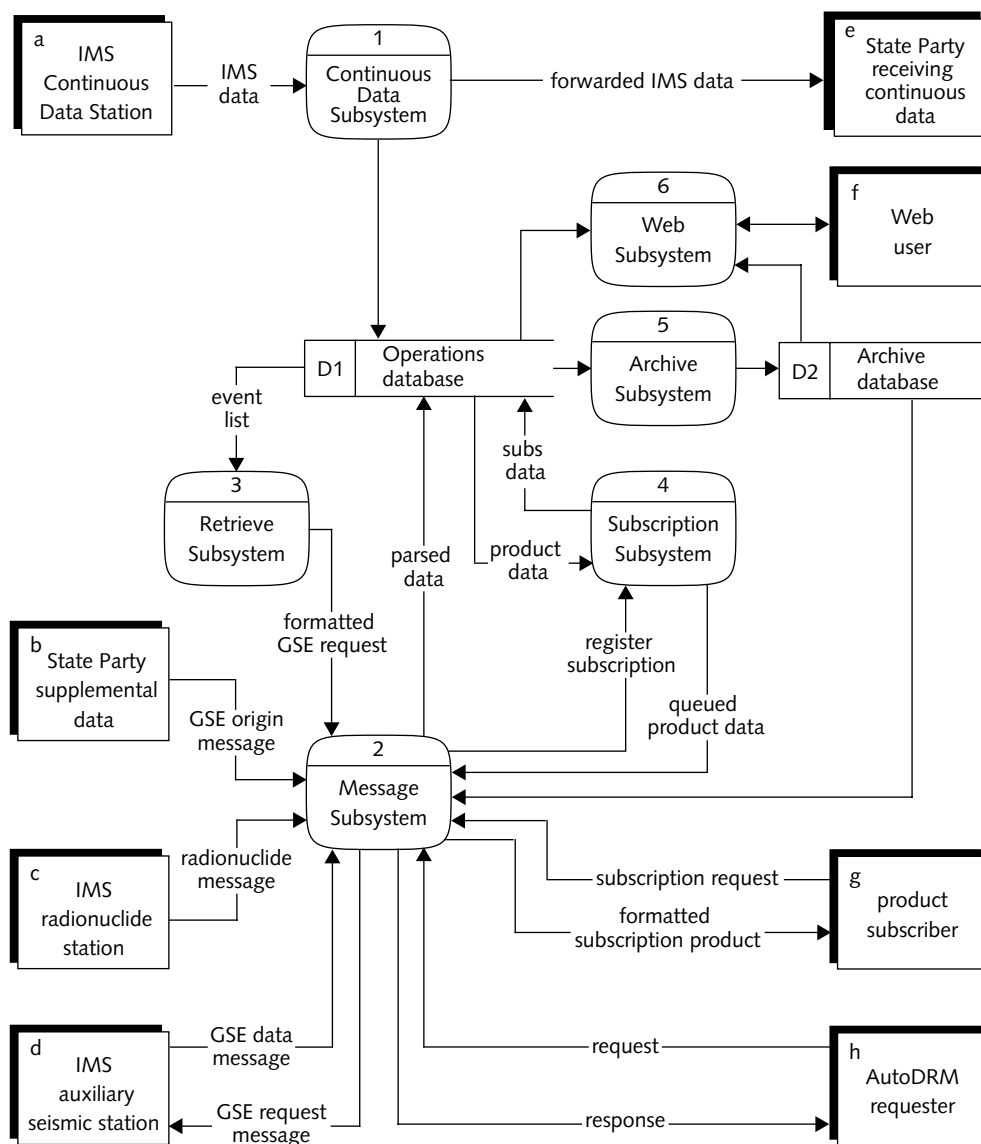


FIGURE 2. RELATIONSHIP OF SUBSCRIPTION SUBSYSTEM TO OTHER SOFTWARE UNITS OF DATA SERVICES CSCI

Overview ▼

Users of the Subscription Subsystem must have access to an SMTP (Simple Mail Transfer Protocol) mail agent. A file transfer protocol (FTP) application is needed to obtain voluminous products. It is also advisable to have software to parse the delivered data.

IDENTIFICATION

The Subscription Subsystem's components are identified in Table 1.

TABLE 1: COMPONENTS OF SUBSCRIPTION SUBSYSTEM

Component	System Version	Component Release Version
<i>DataReady_to_Prodtrack</i>	4.0	Subscription Subsystem Release 4.0
<i>FlatProduct_to_DataReady</i>	1.0	Subscription Subsystem Release 4.0
<i>fpstacap</i>	1.0	Subscription Subsystem Release 4.0
<i>ParseSubs</i>	5.2	Subscription Subsystem Release 4.0
<i>SubsProcess</i>	5.3	Subscription Subsystem Release 4.0
<i>SubsTrack</i>	not released	not implemented
<i>SubsWatch</i>	not released	not implemented
<i>write_fp</i>	1.0	Subscription Subsystem Release 4.0

STATUS OF DEVELOPMENT

The Subscription Subsystem software replaced the previous subscription service software in December 1996. Prior to December 1996, the system was only capable of distributing bulletins daily. The new system supports instant delivery of products (that is, delivery as soon as the data are available) and allows subscriptions to be defined for subsets of available data. Subsets are based on most attributes (including arrivals, detections, and bulletin-associated waveforms) that are recognized by the GSE message protocol (see "Formats and Protocols" in [GSE95b]). The new Subscription Subsystem also adds the ability to track delivery

of the products. In the future, the Subscription Subsystem will support more IDC products, including system status and radionuclide products. In addition, the interface will be expanded to include a Web interface. Also, a graphically-based control and monitoring program will be developed.

BACKGROUND AND HISTORY

David Salzberg and Ed Shnekendorf of the Center for Monitoring Research (CMR) developed the Subscription Subsystem in 1996. The software automatically distributed (based on standing requests) both IDC data and processed data products. The software mailed the subscription either as soon as the product was available or on a regular schedule at the requester's discretion. Standing requests are referred to as subscriptions. For example, a subscriber's request to receive all of the Automated Event List bulletins as soon as they are available is a subscription.

The Subscription Subsystem, first used operationally in December 1996, currently is an element of the IDC at the CMR in Arlington, Virginia, USA. The following additional site is expected to install the Subscription Subsystem software:

The International Data Centre of the Comprehensive Nuclear Test Ban Treaty Organization (CTBTO IDC) in Vienna, Austria.

The Operations staff of the IDC is responsible for operating the Subscription Subsystem. The level of operational involvement is expected to be slight, because the system is designed to run automatically. The software will be maintained by David Salzberg or other developers through the Software Modification Request (SMR) process.

OPERATING ENVIRONMENT

The following paragraphs describe the hardware and commercial-off-the-shelf (COTS) software required to operate the Subscription Subsystem.

Hardware

The Subscription Subsystem software is designed to run on a UNIX workstation such as the SPARCstation 20/712. Typically, the hardware is configured with 64 MB of memory and a minimum of 2 GB of magnetic disk. The required disk space will scale with the duration that outgoing messages are archived. The Subscription Subsystem must obtain other services (such as database access and mail) over its Ethernet connection to other computers. Figure 3 shows a representative hardware configuration.

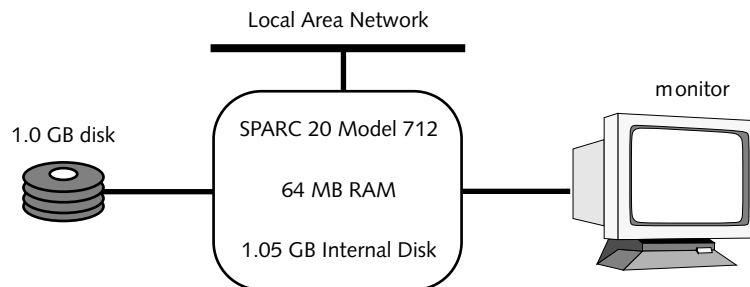


FIGURE 3. REPRESENTATIVE HARDWARE CONFIGURATION FOR SUBSCRIPTION SUBSYSTEM

Commercial-Off-the-Shelf Software

The Subscription Subsystem has been developed under Solaris 2.5 and ORACLE 7.2.3. It has been tested under Solaris 2.6. The software requires access to a TCP/IP mail transfer agent such as *sendmail*.

Architectural Design

This chapter describes the architectural design of the Subscription Subsystem and includes the following topics:

- Conceptual Design
- Design Decisions
- Functional Description
- Interface Design

Architectural Design

CONCEPTUAL DESIGN

The Subscription Subsystem is designed to interact with the Message Subsystem and cannot function independently. The Subscription Subsystem adds automation to the Message Subsystem by providing customized data products to users on a routine basis. The Subscription Subsystem maintains subscriber profiles. As data become available, they are matched against subscriber orders. The Subscription Subsystem then constructs the directives for filling orders and passes this information to the Message Subsystem.

Figure 4 shows an object model of the Subscription Subsystem. Subscribers place subscriptions by defining a subscription object, that is, a set of criteria for the content and delivery of an IDC product. At the time a subscription is to be processed, the filter creates a formal description of the product, and this description is placed into a queue. The software of the Message Subsystem (*AutoDRM*) assembles and formats the data into the final product and delivers the product to the subscriber.

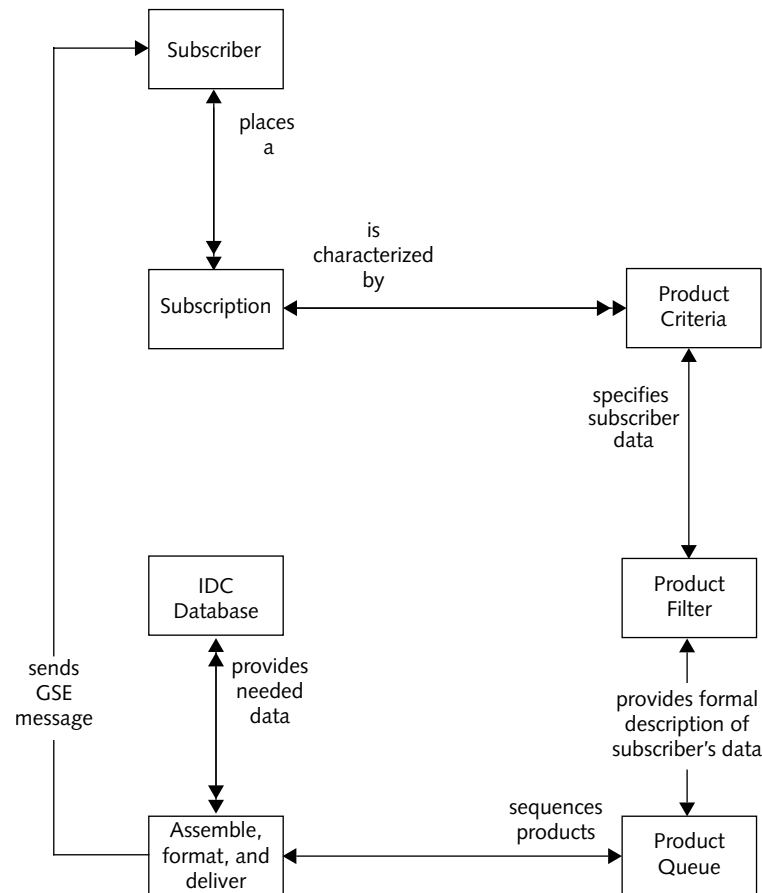


FIGURE 4. OBJECT MODEL OF SUBSCRIPTION SUBSYSTEM

Subscribers may place subscriptions either by email or (in the future) with a form-based Web interface. Email messages must adhere to the standard protocol (see "Formats and Protocols" in [GSE95b]).

Subscriptions are user-specific, routine requests for data. Table 2 indicates the list of products currently supported by the Subscription Subsystem. In addition, a variety of criteria can be invoked to create subscriptions based on non-standard products. This feature allows a subscription to be placed either for a standard IDC data

product or for a product tailored to the subscriber's criteria. Tailoring criteria for each product are also indicated in Table 2.

Many products are created at the IDC in the course of the daily analysis of IMS data. The processing flow depicted in Figure 5 illustrates some of the stages at which products become available for seismic, hydroacoustic, and infrasonic stations of the IMS. Additional products, for example those pertaining to IMS radionuclide data, will become available through the Subscription Subsystem in the future.

The left column of Figure 5 shows the processing flow, with time increasing downwards. At the indicated junctures, products shown in the central column become available for distribution to subscribers. The column on the right indicates when and what action initializes the processing of a subscription.

Three general categories of products are recognized. Detection products have information about signals present in the raw timeseries. Origin products have information about the location and time of events that excited the signals. Waveform products are the underlying timeseries themselves. Origin products are further categorized as origins of Standard Event List 1 (SEL1), origins of Standard Event List 2 (SEL2), and origins of the Reviewed Event Bulletin (REB).

Table 3 describes the terms and abbreviations that appear in the product names.

TABLE 2: STANDARD PRODUCTS AVAILABLE THROUGH SUBSCRIPTION SUBSYSTEM

Product	Availability Time (GMT ¹)	Tailoring Criteria
Instant Detections	~ 10 minutes	Station, channel
Daily Detections	midnight	Station, channel
Instant ² SEL1 Arrivals	~ 20 minutes	Station, channel
Instant ² SEL1 Origins	~ 20 minutes	Hypocenter of event, magnitude
Instant ² SEL1 Bulletin	~ 20 minutes	Hypocenter of event, magnitude

TABLE 2: STANDARD PRODUCTS AVAILABLE THROUGH SUBSCRIPTION SUBSYSTEM (CONTINUED)

Product	Availability Time (GMT ¹)	Tailoring Criteria
Daily SEL1 Arrivals	0100	Station, channel
Daily SEL1 Origins	0100	Hypocenter of event, magnitude
Daily SEL1 Bulletin	0100	Hypocenter of event, magnitude
Instant ³ SEL2 Arrivals	~ 20 minutes	Station, channel
Instant ³ SEL2 Origins	~ 20 minutes	Hypocenter of event, magnitude
Instant ³ SEL2 Bulletin	~ 20 minutes	Hypocenter of event, magnitude
Daily SEL2 Arrivals	0400	Station, channel
Daily SEL2 Origins	0400	Hypocenter of event, magnitude
Daily SEL2 Bulletin	0400	Hypocenter of event, magnitude
Daily ⁴ REB Arrivals	~ daily	Station, channel
Daily ⁴ REB Origins	~ daily	Hypocenter of event, magnitude
Daily ⁴ REB Bulletin	~ daily	Hypocenter of event, magnitude
Instant Auxiliary Waveforms	every minute	Station, channel
Waveform Segments from Events	~ daily ⁵	Hypocenter of event, magnitude
Station Status	~ daily	Station, channel
Channel Status	~ daily	Station, channel

1. Greenwich Mean Time
2. Instant means approximately one hour behind real time.
3. Instant means approximately four hours behind real time.
4. Daily means after analysis is complete.
5. Daily means when segment archive is generated.

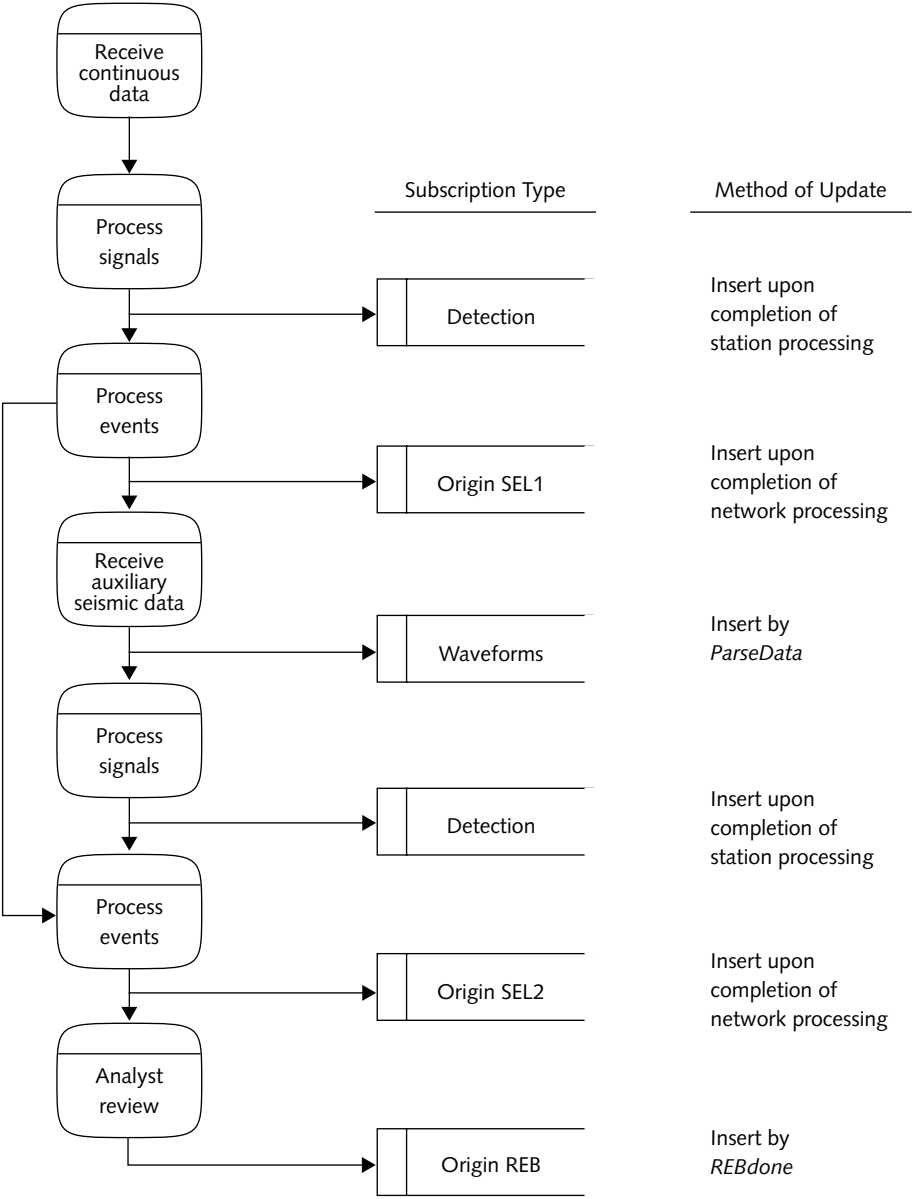


FIGURE 5. PROCESSING FLOW AND CREATION OF DATA PRODUCTS

TABLE 3: TERMS RELATING TO STANDARD PRODUCTS

Name	Description
detection	A detection is a probable signal that has been automatically detected by the Detection and Feature Extraction (DFX) software.
arrival	An arrival is a signal that has been associated to an event. In the first instance, this is performed automatically by the Global Association (GA) software. Later in the processing, many arrivals are confirmed and improved by visual inspection.
origin	An origin is the place and time of a seismic, hydroacoustic, or infrasonic event.
bulletin	A bulletin is a chronological listing of event origins spanning an interval of time. The specification of each origin or event often is accompanied by the event's arrivals and sometimes with the event's waveforms.
SEL1	Standard Event List 1 is the bulletin created by totally automatic analysis of continuous timeseries data. Typically, the list runs one hour behind real time.
SEL2	Standard Event List 2 is the bulletin created by totally automatic analysis of both continuous data and segments of data specifically downloaded from stations of the auxiliary seismic network. Typically, the list runs four hours behind real time.
REB	Reviewed Event Bulletin is the bulletin formed of all events that have passed analyst inspection and quality assurance review. The REB runs 48 hours behind real time.

DESIGN DECISIONS

The following design decisions pertain to the Subscription Subsystem.

Programming Language

Each software unit of the Subscription Subsystem is written in the C programming language unless otherwise noted in this document.

Global Libraries

The software of the Subscription Subsystem is linked to the following shared (developmental) libraries: *libdrm*, *liblogout*, *libpar*, *libtime*, *libinterp*, *libaesir*, and *libgdi*.

The software is also linked to the following COTS libraries: *libsql*, *libsqlnet*, *libncr*, *libclient*, *libcommon*, *libgeneric*, *libepc*, *libnlsrtl3*, *libc3v6*, *libcore3*, *libm*, *libF77*, and *libdl*.

Database

The Subscription Subsystem maintains all key information in an ORACLE database. Numerous tables in the database are used only by the Subscription Subsystem. These tables record information about subscription requests and the state of products before the products are mailed to subscribers. Database triggers are used to initiate product processing.

The criteria by which each product can be tailored are recorded in product-specific criteria tables. This approach entails a more complex schema than using highly normalized tables. An alternative design approach using a single, simpler table for all criteria was rejected. The alternative approach would have entailed significantly more database queries and would have prevented the use of database triggers.

Interprocess Communication (IPC)

The Subscription Subsystem does not use the message services of the Distributed Application Control System (DACS).

File System

The file system holds the run-time parameters of the Subscription Subsystem (parameters files). Certain intermediate data are temporarily stored in the file system. For example, subscription requests are passed from the Message Subsystem to the Subscription Subsystem via a disk file.

Log files are written by two processes, *ParseSubs* and *SubsProcess*.

UNIX Mail

Subscriptions are delivered by UNIX mail, which is managed by the Message Subsystem. Subscription requests are also received via the Message Subsystem through UNIX mail. Users can place requests and receive products through any SMTP (Simple Mail Transfer Protocol) mail agent.

FTP

Products that are too large for reliable mail delivery are posted to an FTP directory; subscribers are notified of the products' availability. This practice follows the normal practice of the Message Subsystem.

Web

An HTML form (to be developed later) will allow users to register subscriptions and check on delivery status. For security reasons, the output of the form will be sent to the Message Subsystem via UNIX mail, and the response will be delivered back via the same mechanism.

Design Model

The design of the Subscription Subsystem is primarily influenced by timeliness, flexibility, and reliability requirements. Although the system must deliver the products in a timely manner, a limited delay is acceptable to allow for interprocess communications (IPC). Reliability and long-term tracking of the subscriptions are also critical requirements. Thus, the system uses the IDC's database for both IPC and to archive the processing information. The implementation additionally allows the products to be customized.

Database Schema Overview

The Subscription Subsystem uses the ORACLE database for the following purposes:

- to record facts about a subscription
- to initiate product creation
- to record the progress of product distribution

Table 4 shows the tables used by the Subscription Subsystem. The Name field identifies the database table. The Mode field is "R" if the Subscription Subsystem reads from the table and "W" if the system writes to the table. The Owner field indicates who has administrative control over the table. The "DBA" (database administrator) owns standard tables that are part of the core schema. The "SS" (Subscription Subsystem) owns tables that are private to the Subscription Subsystem and are not accessed by any other software. A few tables are owned by the Message Subsystem, as indicated by "MSG."

TABLE 4: DATABASE TABLES USED BY SUBSCRIPTION SUBSYSTEM

Name	Mode	Owner	Description
dataready	R/W	SS	contains data needed to process a subscribed product
flatproduct	R/W	SS	each row describes a flatproduct file
flatdescription	R/W	SS	description of flatproduct types
prodtrack	R/W	SS	tracks delivery of products to subscribers
productcriteria	R/W	SS	records characteristics of standard and custom products
producttype(*)	R/W	SS	additional tables for other products will be defined
producttypeorigin	R/W	SS	each row describes a customized origin or bulletin subscription
productypesta	R/W	SS	each row describes a customized detection or arrival subscription
subs	R/W	SS	records email addresses of subscribers
subsuser	R/W	SS	records qualified users of Subscription Subsystem
msgdest	R/W	MSG	records destinations for outgoing messages
msgdisc	R/W	MSG	contains pointers to message files for both incoming and outgoing messages
arrival	R	DBA	contains data on a signal that has been identified as an arrival
assoc	R	DBA	contains data that relate arrivals to origins
detection	R	DBA	contains information on a detected signal
origin	R	DBA	describes an origin (location, magnitude, and so on)
origerr	R	DBA	contains uncertainties/errors in the origins
site	R	DBA	contains data on a station site, such as latitude, longitude, and elevation
sitechan	R	DBA	contains channel specific information on a site, such as channel orientation and sample rate

FUNCTIONAL DESCRIPTION

The Subscription Subsystem consists of two main functions: a function for parsing and registering subscriptions and a function for preparing and defining products. These functions are shown in Figure 6 (for subscription parsing) and Figure 7 (for product preparation and distribution). Additionally, the latter figure shows the interfaces between the Subscription Subsystem and the Message Subsystem and between the Subscription Subsystem and the data processing function.

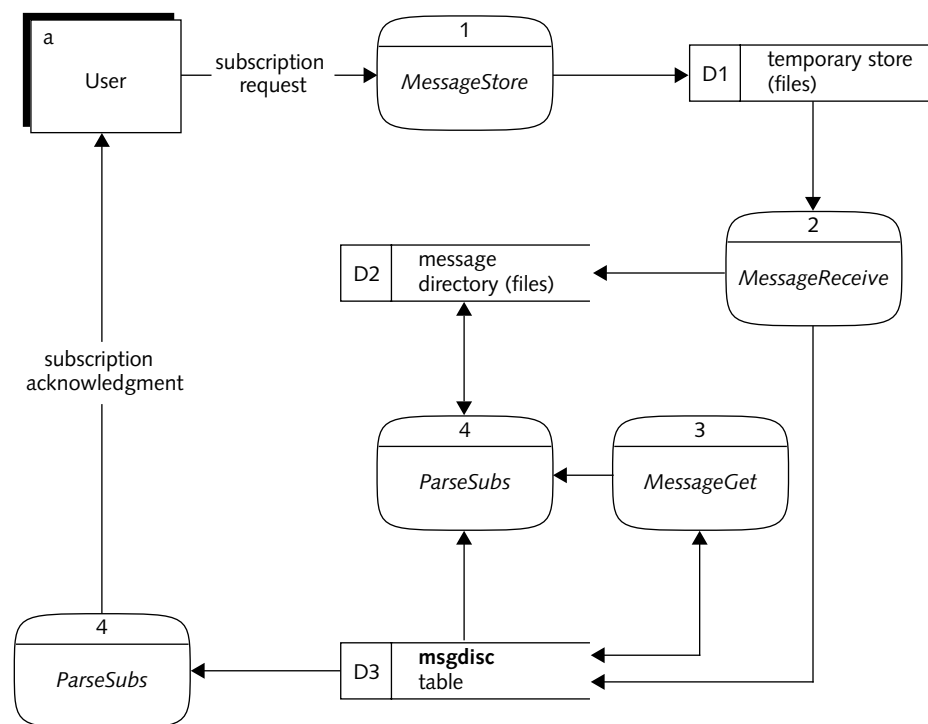


FIGURE 6. FLOW OF SUBSCRIPTION PARSING

Subscription requests (submitted by email) are recognized as such by the Message Subsystem. The requests are relayed to the parsing module of the Message Subsystem (Figure 6, process 4, and Figure 7, process 2), which registers the request by adding information to several ORACLE tables.

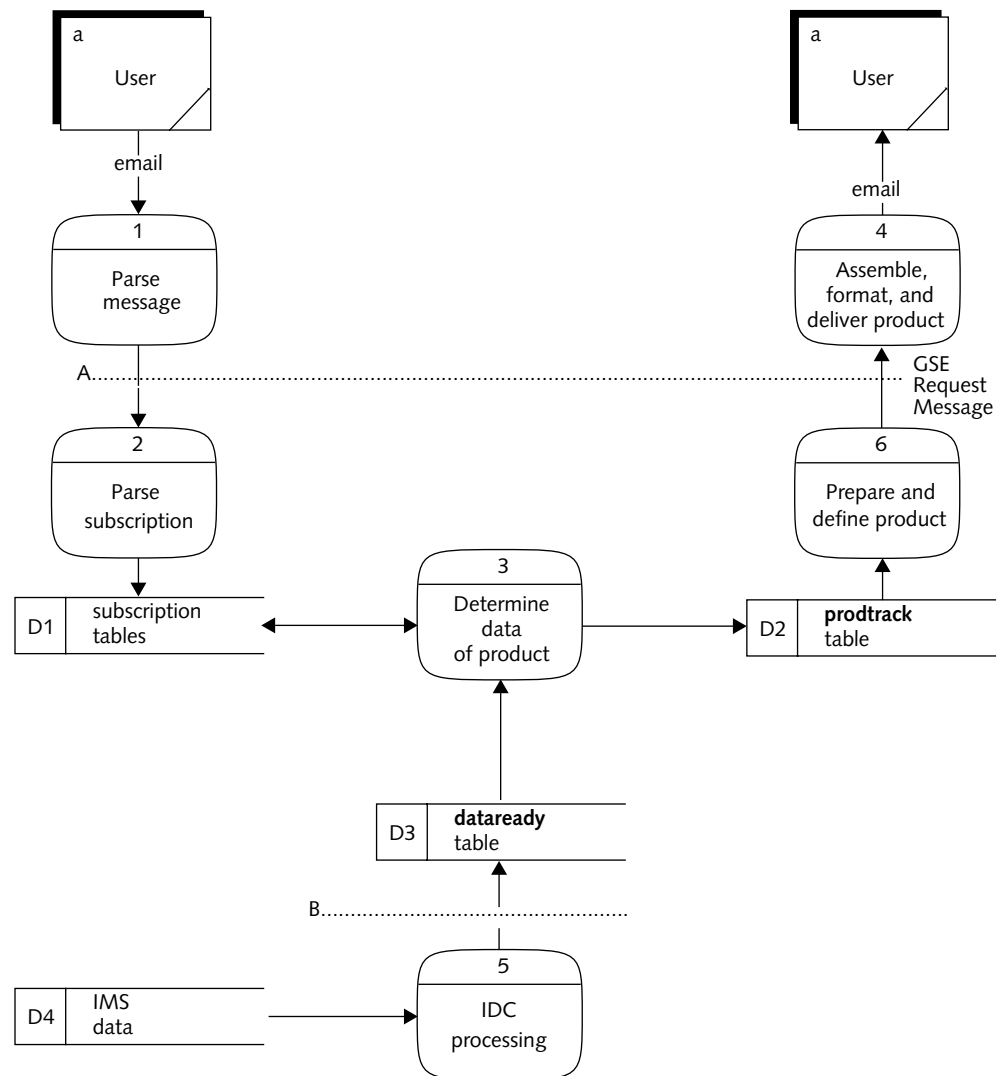


FIGURE 7. DATA FLOW AND EXTERNAL INTERFACES OF SUBSCRIPTION SUBSYSTEM

The availability of data products is advertised by the regular insertion of entries in the **dataready** table. These notifications occur automatically at various stages of the data processing chain displayed previously in Figure 5. The table acts as an inter-

face between the Subscription Subsystem and the automatic and interactive processing subsystems.

The data pointed to by rows in **dataready** are filtered according to product criteria established by subscription parsing (Figure 7, process 3). The filtered data are inserted into the **prodtrack** table, which holds information used for tracking products in the queue and products mailed for delivery. The **prodtrack** table is periodically polled by the subscription processing program, *SubsProcess*, which starts a formatting process through the Automated Data Request Manager, *AutoDRM* (Figure 7, process 4), a unit of the Message Subsystem.

Receiving Subscription Requests

The Subscription Subsystem's user interface is either an electronic mail message (email) or a form on the Web that is converted to email. The Message Subsystem recognizes from information in the email that the message is to be handled by the Subscription Subsystem. Figure 6 illustrates the details of this process. The subscriber sends an email message to the address `messages@pidc.org`. The message is temporarily stored by the procedure *MessageStore*. *MessageReceive* polls the temporary storage location for new messages, moves the messages to a staging area, creates a new row in the message description table, **msgdisc**, and then moves the message file to a permanent location (Figure 6, D2, message directory). *MessageGet* polls the **msgdisc** table for newly received rows and then starts a process to handle the message. If the message type is SUBSCRIPTION, then the process *ParseSubs* is started by *MessageGet*.

Registering Subscriptions

ParseSubs (Figure 6, process 4) reads the flat file containing the email message, which must be formatted according to the rules explained in [IDC3.4.1]. *ParseSubs* updates the subscription database tables if the request was to create a new subscription, change an existing subscription, or delete a subscription. *ParseSubs* then mails back a response informing the requester of the subscription maintenance action.

Processing and Formatting Subscriptions

The Subscription Subsystem begins processing and formatting subscriptions when a row is inserted into the **dataready** table (Figure 7, D3), advertising that certain data are ready. The row is inserted by operators of the IDC or by software used by the operators to manage the IDC. Upon inserting the row, a database trigger launches *DataReady_to_ProdTrack*, which compares the new data to the products in the **productcriteria** and **producttype(*)** tables. A row is inserted into the **prodtrack** table for each defined product that must incorporate the new data. This feature allows the product to be queued, processed, and tracked.

SubsProcess regularly checks the **prodtrack** table for new rows. This action is initialized by an internal timer in the UNIX system (using the C function `sleep` in *SubsProcess*). Each time *SubsProcess* wakes, it scans the **prodtrack** table for new rows. Each new row is processed to yield one or more GSE Request Messages. Each message contains the formal description of a product, according to specifications in the **productcriteria** and **producttype(*)** tables, and the data as defined in the **dataready** table. The outgoing product is then created and delivered using elements of the Message Subsystem, *AutoDRM*, and message delivery program, *MessageShip*.

Error Handling

Error handling is integral to the design of the Subscription Subsystem. All of the processes trap signals in case unexpected problems occur in the software. In addition, the *status* field of the **msgdisc** (for parsing of subscriptions) and **prodtrack** (for processing of the products) tables are designed for easy recovery after failures. The parsing of subscriptions also generates an error log that is returned to the requester. The log contains a list of the problems that occurred while parsing the subscription request. If failures occur during subscription processing, the value of **prodtrack.status** is set to **FAILED**. To recover from the failure of subscription processing, the value of **prodtrack.status** is updated to **NEW**, then the product is re-queued.

Tracking Subscriptions

The Subscription Subsystem uses various columns of the **prodtrack** table to track subscription registration messages, delivered products, and the data used to generate the products. The columns include the product identifier (*prodid*), the delivery identifier for each product (*delivid*), the message identifier for the outgoing product (*msgid*), and the identifier for the data description (*dataid*). By combining the subscription and the user name, the products can be tracked to the subscription request message or to all products that are to be delivered to a subscriber. By joining the tables, an individual data point can be linked to all of the product deliveries that contain that point.

INTERFACE DESIGN

This section describes how the Subscription Subsystem interfaces with other IDC systems, external users, and operators.

Interface with Other IDC Systems

The principal interfaces between the Subscription Subsystem and other software were indicated in Figure 7. The top of the figure shows two interfaces with the Message Subsystem. Subscription registration requests pass from the message parser to the subscription parser through the left interface. The exchange is mediated by the **msgdisc** table. Specifically, the attributes **msgdisc.status** and **msgdisc.msgtype** indicate the proper instruction for handling the message.

Instructions for preparing and delivering data are passed to the Message Subsystem through the top-right interface. This exchange is mediated by the **msgdisc** and **msgdest** tables. The attribute **msgdest.status** indicates the message is ready to be delivered to **msgdest.emailto**.

The interface between the Subscription Subsystem and the entire IDC processing enterprise is indicated towards the bottom of the figure. The interface consists of the Subscription Subsystem's **dataready** table, to which rows are added by numerous IDC functions.

Interface with External Users

Users interact with the Subscription Subsystem by submitting properly formatted mail messages or by completing a Web form. In either case, the formatted message is forwarded to the Message Subsystem and passed off to the Subscription Subsystem, as described above. The format of messages is described in "Formats and Protocols" in [GSE95b]. The Subscription Subsystem verifies that the subscriber is authorized to use the system by comparing the subscriber's email address to information stored in a database table. The Subscription Subsystem then reads the subscription request message. The Message Subsystem delivers a response message to the user to confirm the subscription maintenance action.

Interface with Operators

The Subscription Subsystem is built around the database; therefore, the interface with the operators of the system is built around SQL statements. Operators submit specific SQL queries for required maintenance. Although SQL-Plus could support the entire system, coding the interface would be difficult. A Graphical User Interface (GUI) tool will be developed to simplify the operators' tasks of monitoring and maintaining the Subscription Subsystem.

Detailed Design

This chapter describes the detailed design of the Subscription Subsystem and includes the following topics:

- Data Flow Model
- Software Units
- Database Description

Detailed Design

DATA FLOW MODEL

The new subscription service is based on a series of objects (see Figure 4). Data objects contain information that corresponds to the subscription, data, and outgoing product. Processing objects prepare and distribute certain data objects. The following associations connect the objects: subscriber requests a subscription(s). The subscription points to a set of product criteria (many subscriptions can point to a single product). As products become available, they are filtered by the product criteria. The results of the filtering are GSE Request Messages, which are queued.

The objects that form the basis of a subscription correspond to database tables. The data are passed to the tables through a variety of mechanisms. The information regarding the subscription request is added by the procedure that parses subscription requests, *ParseSubs* (Figure 7, process 2).

Product delivery is initiated through the database table **dataready**, which is updated by pipeline operations when products are ready to be distributed. Each row of the table points to a specific product. The *DataReady_to_ProdTrack* process (Figure 7, process 3) compares the rows in the **dataready** table with all possible product criteria. Each match is noted by adding a row to **dataready**, the product queuing and tracking table. The **prodtrack** table contains one entry for each subscription-product pair. For example, if one origin matches the criteria for five products, then five entries are inserted into the **prodtrack** table.

The formatting and processing software, *SubsProcess* (Figure 7, process 4), queries the **prodtrack** table for new entries. When *SubsProcess* finds new entries, it uses all of the detailed criteria of the pertinent subscription to construct a proper GSE Request Message.

The system will track subscriptions and handle errors by linking the outgoing messages to the **subs**, **productcriteria**, and **dataready** tables. The **prodtrack** table will contain the message identifier (*msgid*) for the outgoing message, the data ready identifier (*dataid*), the subscription identifier (*subsid*), and the product identifier (*prodid*).

Daily subscriptions are processed at the end of a data-day or processing-day; immediate subscriptions are processed as soon as the underlying product is available. Thus, an immediate subscription will allow a subscriber to receive data as quickly as possible after each stage of processing. A daily subscription will contain the same information as the immediate subscription. In addition, any product may contain updates and deletions if needed. The updates and deletions will be sent as `DATATYPE LOG`, a type recognized in the message protocol (see “Formats and Protocols” in [GSE95b]).

SOFTWARE UNITS

The Subscription Subsystem consists of the following software units:

- *DataReady_to_ProdTrack*
- *FlatProduct_to_DataReady*
- *fpstacap*
- *ParseSubs*
- *SubsProcess*
- *SubsTrack (not implemented)*
- *SubsWatch (not implemented)*
- *write_fp*

The following paragraphs describe the design of these units, including any constraints or unusual features in the design. The logic of the software and any applicable procedural commands are also provided.

DataReady_to_Prodtrack

DataReady_to_ProdTrack, a PL/SQL module, determines which data goes to which product. *DataReady_to_ProdTrack* is triggered by insertion of a new row into the **dataready** table. One row is inserted into the **prodtrack** table for each product that is to be generated from the data. The procedure sets the *prodid*, *dataid*, *status*, and *lddate* attributes. The value of the *prodid* is the *prodid* for the screening **productcriteria**/**producttype** rows. The value for the *dataid* is the new *dataid* in the **dataready** table. The *status* field is set to **NEW**, which indicates (to *SubsProcess*) that a new product is ready to be generated.

Input/Processing/Output

The module's input consists of the new rows in the **dataready** table and the stored rows in the **productcriteria** and **producttype**(*) tables. The output is zero or more new rows in the **prodtrack** table.

Control

DataReady_to_ProdTrack is a database trigger on the **dataready** table. Thus, it is event driven upon the insertion of a row into that table.

Interfaces

DataReady_to_ProdTrack has no user or external (to the Subscription Subsystem) interfaces.

Error States

DataReady_to_ProdTrack has no error-catching mechanism. The program will fail if the database fails, which would preclude the trigger from running or cause it to cease in mid-operation. The program will also fail if required tables have been dropped.

FlatProduct_to_DataReady

The *FlatProduct_to_DataReady* software unit is a database trigger that creates a **dataready** row upon insertion of a **fileproduct** row. By inserting the row into the **dataready** table, the product can then be processed and delivered by the Subscription Subsystem.

Fpstacap

The *fpstacap* module supports file system (as opposed to database tables) products.

ParseSubs

ParseSubs controls the user interface to the Subscription Subsystem. Its tasks are to parse a GSE 2.0-formatted subscription request and to perform the necessary actions, such as creating new database rows or creating and distributing logs of subscription actions. *ParseSubs* obtains information about the message, including its file name, from the database. *ParseSubs* next queries the **subsuser** table both to confirm that the user is authorized to use the Subscription Subsystem and to determine the *userid* by comparing the account name and domain name in the email header to those stored in the **subsuser** table. If the user is authorized, the message file is opened and parsed one line at a time. The GSE 2.0 subscription environment lines are stored in global structures to be recalled when a command line is reached (see “Formats and Protocols” in [GSE95b]). When the command line of the message is reached, the system calls a function that is specific to the parsing of the stated command. The Subscription Subsystem indicates that a command line is reached by the key words BULLETIN, ARRIVAL, SUBSCR_LOG, and so on (see “Formats and Protocols” in [GSE95b]). The function first retrieves the relevant environment variables; if a particular value is undefined, the function uses default values, unless the environment is required. If a mandatory environment variable is missing, a parsing error is returned to the user. *ParseSubs* performs the requested action, creating the new rows in the subscription tables or retrieving the

rows for a log or other message. *ParseSubs* then generates a reply message informing the user of the action.

Input/Processing/Output

ParseSubs requires the following inputs:

- the pointer to the message file (**msgdisc.msgid**)
- the message file itself
- information about valid users contained in the **subsuser** database table

ParseSubs then generates new rows in the database tables **subs** and **productcriteria** and returns a message to the requester.

Control

ParseSubs is started automatically by the *MessageGet* process and terminated automatically after the message has been processed.

Interfaces

ParseSubs interfaces with the Message Subsystem, as described previously, with the database, and with the file system. *ParseSubs* has no important internal interfaces, other than the application programming interfaces to the various software libraries.

Error States

ParseSubs performs extensive checks on the message, which limits the possibility that the subsystem will fail from erroneous subscription request data. If the program cannot connect to the ORACLE database, the program terminates with a nonzero exit status, and the parent process (*MessageGet*) cleans up. In the event of a UNIX crash, the value of **msgdisc.status** indicates that its work was not finished; the operator will manually relaunch the program.

SubsProcess

SubsProcess oversees the creation of the data products by preparing necessary instructions for assembling, formatting, and delivering by the Message Subsystem. *SubsProcess* constructs a GSE Request Message for each product. The module then invokes *AutoDRM* to interpret the messages, retrieve the data from the IDC database, and format the data into GSE messages of the appropriate type. These product messages (or files) are then delivered to subscribers through the software of the Message Subsystem.

First, *SubsProcess* searches the **prodtrack** table for records in which attribute *status* has the value **NEW**. In such records, the field is changed to the value **QUEUED**. This effectively locks the table in a known state and permits additional records of this type to be added concurrently as *SubsProcess* is executing.

Next, *SubsProcess* reads all of the rows that have a status of **QUEUED**. This action also picks up products that were not generated due to a previous software failure. *SubsProcess* then processes the queued rows one at a time. As each product is being prepared, **prodtrack.status** is updated to **RUNNING**. *SubsProcess* then reads in the product tables (**productcriteria** and the appropriate **producttype** table). Based on those criteria, a standard GSE Request Message is constructed. In addition, the product associated with each Request Message is assigned a unique, sequential delivery identifier, *delivid*. After the Request Message is prepared, the message is written to a file, accompanied by a pointer in the message descriptor table, **msgdisc**. In addition, the value of **prodtrack.msgid** is set to the value of **msgdisc.msgid** for the new product, and the value of **prodtrack.status** is updated to **DONE**, unless any part of the procedure fails, in which case **prodtrack.status** is set to **FAILED**.

In its final action, *SubsProcess* executes the program *AutoDRM*, passing it the pointers to all of the Request Messages. *AutoDRM* subsequently reads the request messages, retrieves the data, formats the products, and writes them to files.

SubsProcess captures the exit status of *AutoDRM* and sets the value of **prodtrack.status** as appropriate.

Input/Output

When the program is first launched, configuration information, such as the sleep time, is provided either on the command line or from a par file. Each time the program awakes, it reads the table **prodtrack** to formulate its work plan. The **product-criteria** table and various **producttype(*)** tables also provide information to *SubsProcess*. The program also receives status information from *AutoDRM*.

The outputs of *SubsProcess* are files holding GSE Request Messages, as well as new rows in various database tables that indicate the existence and status of the Request Messages.

Control

SubsProcess is designed to run continuously. It is normally initiated in the typical UNIX manner from the run command script that is executed when the host machine boots. It can also be started by an operator. After completing the current cycle of processing the queue, *SubsProcess* sleeps for a defined time. Upon awakening, it examines the **prodtrack** table for new work.

Interfaces

SubsProcess interfaces with the Message Subsystem and with the database, as described previously. *SubsProcess* has no important internal interfaces, other than the application programming interfaces to the various software libraries.

Error States

If the program cannot connect to the ORACLE database, the program terminates with a nonzero exit status. This error is recoverable because all work remains in the queue. If the program cannot fork *AutoDRM*, it returns with the value of *status* set to **FAILED**. All work is requeued so it can be processed at the next opportunity.

SubsTrack

This software unit, which has not been developed, is intended to track subscriptions by reading **prodtrack**, **msgdisc**, and **msgdest** to determine which table received which data. This software unit may be incorporated into *SubsWatch*.

SubsWatch

This “watchdog” software unit, which has not been developed, is intended to verify that all products are sent in a timely manner (as defined by a run-time parameter). This program will also notify operators of any failures.

Write_fp

The *write_fp* module supports file system (as opposed to database tables) products.

DATABASE DESCRIPTION

The Subscription System relies extensively on the database to store, track, and process the subscriptions. All interaction with the database is through the Generic Database Interface (GDI) [And94]. This section describes the Database Design, Database Entity-Relationship Model, and Database Schema.

Database Design

The Subscription Subsystem uses a database for storing, tracking, and queuing products. The entity-relationship model of the schema is indicated in Figure 8. The user, as defined by a row in the **subsuser** table, subscribes to a particular product. The subscriptions are defined by rows in the **subs** table and are joined to the product definition tables by the product identifier, *prodid*. Users can subscribe to an unlimited number of products, and unlimited users may subscribe to a particular product. The products are defined by the **productcriteria** table. The type of product is identified by the *prodtype* attribute in the **productcriteria** table. The **producttype-**

origin table defines constraints for origin-based products; The **producttypesta** table defines constraints for station-based products. In some cases, both tables define constraints for a product. In addition, an individual **productcriteria** can have multiple sets of constraints, such as multiple stations defined in **producttypesta** or multiple origin bounding boxes defined in **producttypeorigin**. The **prodtrack** table links the product definitions to the product deliveries and joins them to the data tables (**dataready**) and the Message Subsystem tables through **prodtrack.msgid**.

The relatively static tables, **subsuser**, **subs**, **productcriteria**, **producttypesta**, and **producttypeorigin** will each remain small, containing less than a few hundred rows. The **dataready** table will add one data row each time a product becomes available. The net growth in the table will be approximately 10,000 rows/day. The **prodtrack** table will grow at a similar rate.

The Subscription Subsystem will have minimal impact on the database performance. The **prodtrack** table will receive three updates for each row; one update will be read to the **dataready**, **productcriteria**, and **subs** table for each row in the **prodtrack** table.

Database Schema

The Subscription Subsystem uses the following tables, which are detailed below:

- **dataready**
- **fpdescription**
- **fileproduct**
- **prodtrack**
- **productcriteria**
- **producttype(*)**
- **subs**
- **subsuser**

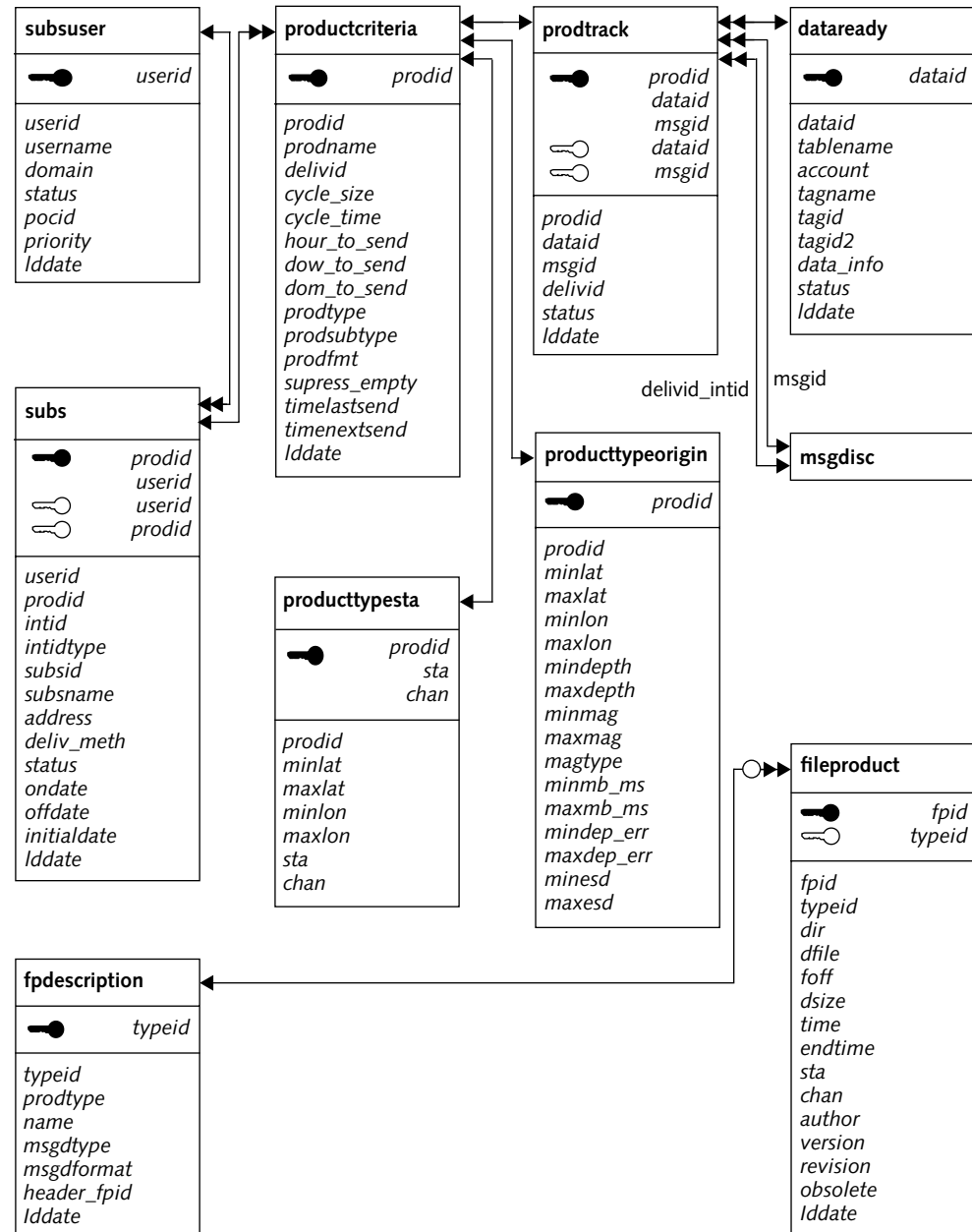


FIGURE 8. ENTITY-RELATIONSHIP MODEL OF SUBSCRIPTION SUBSYSTEM TABLES

Dataready

The **dataready** table contains the basic information required to process a subscription request after a product becomes available. The table contains the local *dataid*, the table and account that contain the product, the reference tagname, the value for that account, the type (or status) of the new entry, and the load date. Table 5 shows the **dataready** table. Primary and foreign keys are identified in Figure 8.

TABLE 5: DATAREADY

Column	Storage Type	Description
<i>dataid</i>	number(8)	identifier for the dataready table, set on the insertion
<i>tablename</i>	varchar2(24)	name of the table that has changed
<i>account</i>	varchar2(24)	account that has changed
<i>tagname</i>	varchar2(12)	name of the reference field in <i>tablename</i> (for example, <i>arid</i> for arrivals)
<i>tagid</i>	number(10)	value of <i>tagname</i> that changes
<i>tagid2</i>	number(10)	secondary <i>tagid</i> (particularly useful for time ranges)
<i>data_info</i>	varchar2(24)	miscellaneous data information
<i>status</i>	varchar2(8)	status of the new entry: (i)nsert, (c)hange, (d)elele
<i>lddate</i>	date	load date of the new entry

Fpdescription

The **fpdescription** table describes the product types available as files. The table is used by software that is under development.

TABLE 6: FPDESCRIPTION

Column	Storage Type	Description
<i>typeid</i>	number(8)	identifier for the product type description
<i>prodtype</i>	varchar2(12)	name of the product (should be the same as the protocol defining name)
<i>name</i>	varchar2(64)	descriptive listing of the product name
<i>msgdtype</i>	varchar2(16)	type of data (for example, ASCII, GIF89, and so on)
<i>msgdformat</i>	varchar2(8)	format of the data (GSE 2.0, RMS 1.0, and so on)
<i>header_fpid</i>	number(8)	<i>fpid</i> pointing to the header row for this product type
<i>lddate</i>	date	load date

Fileproduct

The **fileproduct** table describes a file holding a data product. This table is used by software that is under development.

TABLE 7: FILEPRODUCT

Column	Storage Type	Description
<i>fpid</i>	number(8)	flat product identifier
<i>typeid</i>	number(8)	identifier for the product type
<i>dir</i>	varchar2(64)	directory where the flat product is stored
<i>dfile</i>	varchar2(32)	file name containing the flat product
<i>foff</i>	number(8)	file offset (in bytes) where the data begins
<i>dsize</i>	number(8)	size of the data (in bytes)
<i>time</i>	float(53)	beginning time of the data
<i>endtime</i>	float(53)	ending time of the data
<i>sta</i>	varchar2(6)	station name for the data
<i>chan</i>	varchar2(8)	channel name for the data
<i>author</i>	varchar2(16)	author of the flat product
<i>version</i>	float(8)	version of the author
<i>revision</i>	number(4)	revision number of the flat product (in case it is rerun)
<i>obsolete</i>	number(1)	flag to indicate if the data in the file are obsolete
<i>lddate</i>	data	load date

Prodtrack

This table holds the information required to track an individual subscription product to a particular outgoing message. Each row of the **prodtrack** table indicates which subscriber gets the product referenced by the particular *dataid*. The **prodtrack** table is then the basis from which the GSE Request Message is constructed. In addition, this table tracks which product received which *tagid*. In the event of a change or deletion, the table records who received the original value, allowing only the proper recipient(s) to be notified of the modification. Primary and foreign keys are identified in Figure 8.

TABLE 8: PRODTRACK

Column	Storage Type	Description
<i>prodid</i>	number(8)	identifier for the outgoing product
<i>dataid</i>	number(8)	identifier for the dataready table
<i>msgid</i>	number(8)	identifier of the outgoing msgdisc entry
<i>delivid</i>	number(8)	identifier for each product, which increases uniformly with each delivery
<i>status</i>	varchar2(12)	status of the product
<i>lddate</i>	date	load date of the product

Productcriteria

The **productcriteria** table provides details for products, such as the timing of the deliveries and the product type. Table 9 shows the **productcriteria** table. Primary and foreign keys are identified in Figure 8.

TABLE 9: PRODUCTCRITERIA

Column	Storage Type	Description
<i>prodid</i>	number(8)	product identifier
<i>prodname</i>	varchar2(24)	product name (used only for standard subscriptions)
<i>delivid</i>	number(8)	last delivery identifier for this product
<i>cycle_size</i>	number(8)	size to be obtained before transmitting (0 for no limit)
<i>cycle_time</i>	number(4)	cycle time of the subscription (0 for immediate)
<i>hour_to_send</i>	number(2)	time (hours) when the subscription is sent
<i>dow_to_send</i>	number(2)	day of week to send subscription
<i>dom_to_send</i>	number(2)	day of month to send subscription
<i>prodtype</i>	varchar2(12)	product type (origin, detection, and so on)
<i>probsubtype</i>	varchar2(12)	one of the standard lists or bulletins (for example, SEL1, SEL2, REB, and so on)
<i>prodfmt</i>	varchar2(6)	format of the outgoing product (GSE 2.0, HTML, and so on.)
<i>supress_empty</i>	char(2)	suppression of empty messages (Y)es or (N)o
<i>timelastsend</i>	float(54)	epoch time of the last transmission of this product
<i>timenextsend</i>	float(54)	epoch time for the next transmission of this product
<i>lddate</i>	date	load date

Producttype(*) Tables

The **producttype(*)** tables contain the constraint details for each subscription. Two tables of this type are currently defined, **producttypeorigin** and **producttypesta**. Table 10 indicates the generic structure of the **producttype(*)** table. Primary and foreign keys are identified in Figure 8.

TABLE 10: PRODUCTTYPE(*)

Column	Storage Type	Description
<i>prodid</i>	number(8)	product identifier
other constraints available to the subscriber		other product-dependent constraints (for example, minimum latitude, maximum latitude, station name, and so on)

Producttypeorigin

The **producttypeorigin** table has a column for each constraint that possibly can be imposed on an origin product. Table 11 shows the **producttypeorigin** table. Primary and foreign keys are identified in Figure 8.

TABLE 11: PRODUCTTYPEORIGIN

Column	Storage Type	Description ¹
<i>prodid</i>	number(8)	product identifier
<i>minlat</i>	float(24)	minimum latitude of event
<i>maxlat</i>	float(24)	maximum latitude of event
<i>minlon</i>	float(24)	minimum longitude of event
<i>maxlon</i>	float(24)	maximum longitude of event
<i>mindepth</i>	float(24)	minimum depth
<i>maxdepth</i>	float(24)	maximum depth
<i>minmag</i>	float(24)	minimum magnitude
<i>maxmag</i>	float(24)	maximum magnitude
<i>magtype</i>	varchar2(4)	magnitude type (mb, Ms, ML)
<i>minmb_ms</i>	float(24)	minimum value of mb minus Ms
<i>maxmb_ms</i>	float(24)	maximum value of mb minus Ms
<i>mindep_err</i>	float(24)	minimum value of depth minus error
<i>maxdep_err</i>	float(24)	maximum value of depth minus error
<i>minesd</i>	float(24)	minimum event station distance
<i>maxesd</i>	float(24)	maximum event station distance

1. mb = bodywave magnitude, ML = local magnitude, Ms = surface magnitude

Producttypesta

The **producttypesta** table has a column for each constraint that possibly can be imposed on a station-related product. Table 12 shows the **producttypesta** table. Primary and foreign keys are identified in Figure 8.

TABLE 12: PRODUCTTYPESTA

Column	Storage Type	Description
<i>prodid</i>	number(8)	product identifier
<i>minlat</i>	float(24)	minimum latitude of station
<i>maxlat</i>	float(24)	maximum latitude of station
<i>minlon</i>	float(24)	minimum longitude of station
<i>maxlon</i>	float(24)	maximum longitude of station
<i>sta</i>	varchar2(6)	station name
<i>chan</i>	varchar2(8)	channel name

Subs

The **subs** table records the email addresses of subscribers. In addition, the table indicates a user identifier, *userid*; this attribute is the key for joining the table to **subsuser**. Table 13 shows the **subs** table. Primary and foreign keys are identified in Figure 8.

TABLE 13: SUBS

Column	Storage Type	Description
<i>userid</i>	number(8)	user identifier
<i>prodid</i>	number(8)	product identifier
<i>intid</i>	number(8)	internal identifier to track the source of the subscription
<i>intidtype</i>	varchar2(16)	field name of the internal identifier (usually <i>msgid</i>)
<i>subsid</i>	number(8)	subscription identifier
<i>subsname</i>	varchar2(24)	user-supplied name for the subscription
<i>address</i>	varchar2(64)	address to send subscription (email or FTP address)
<i>deliv_meth</i>	varchar2(6)	delivery method (email, FTP)
<i>status</i>	varchar2(6)	inactive (i) or active (a)
<i>ondate</i>	date	on-date for the subscription
<i>offdate</i>	date	off-date for the subscription
<i>initialdate</i>	date	initial date of the subscription
<i>lddate</i>	date	load date

Subsuser

The **subsuser** table tracks authorized users of the IDC Subscription Subsystem. Each user is identified by a (unique) *username* and *domain*, which must match all email headers. The *priority* field is not used by the Subscription System. Primary and foreign keys are identified in Figure 8.

TABLE 14: SUBSUSER

Column	Storage Type	Description
<i>userid</i>	number(8)	user identifier
<i>username</i>	varchar2(24)	user name from the incoming subscription message
<i>domain</i>	varchar2(48)	domain name from the incoming subscription message
<i>status</i>	varchar2(24)	status of this user
<i>pocid</i>	number(8)	point-of-contact identifier
<i>priority</i>	number(2)	priority that this user has at the IDC
<i>lddate</i>	date	load date for the IDC user

Requirements

This chapter describes the requirements of the Subscription Subsystem and includes the following topics:

- Introduction
- General Requirements
- Functional Requirements
- System Requirements
- Requirements Traceability

Requirements

INTRODUCTION

The requirements of the Subscription Subsystem can be categorized as general, functional, or system requirements. General requirements are nonfunctional aspects of the Subscription Subsystem. These requirements express goals, design objectives, and similar constraints that are qualitative properties of the system. The degree to which these requirements are actually met can only be judged qualitatively. Functional requirements describe what the Subscription Subsystem is to do and how it is to do it. System requirements pertain to general constraints, such as compatibility with other IDC subsystems, use of recognized standards for formats and protocols, and incorporation of standard subprogram libraries.

GENERAL REQUIREMENTS

The Subscription Subsystem will meet the following general requirements:

1. The system shall require minimum operator intervention.
2. Impact upon other IDC resources (such as the Database Management System) shall be minimized.
3. The system shall be extensible.
4. The system shall make maximum use of other elements of the Data Services CSCI.
5. The system shall be scalable.

FUNCTIONAL REQUIREMENTS

The requirements described in this section are categorized by function.

Storage of User Profiles and Subscription Records

The Subscription Subsystem is required to store user profiles, contact information, and subscription requests as follows:

6. Users shall be identified by an email address.
7. The system shall associate additional point-of-contact information with each user.
8. The system shall have the ability to associate subscription records with user profiles.
9. The system shall have the ability to authenticate and log a user profile insertion or update.
10. Users shall have the ability to configure an indefinite number of subscriptions.
11. The system shall maintain the date that a subscription was originally configured.
12. The system shall maintain the date that a subscription configuration was last changed.
13. The system shall have the ability to determine whether or not a subscription is active.

Subscription Configuration

The following requirements describe how user subscriptions will be configured:

14. Users shall be able to configure subscriptions through the Web interface.
15. Users shall be able to configure subscriptions through email.
16. The system shall have the ability to authenticate a subscription configuration.

Requirements ▼

17. The subscription may be configured to return data regularly. Regularity is defined as the ability to send data as soon as it is received, daily, weekly, or monthly.
18. Provisions shall be made to distribute data based upon reaching a defined size limit.
19. Provisions shall be made to deliver data, in conjunction with requirements 17. and 18., on a given day, date, and/or time as they are requested for some products.
20. Users may specify a destination/delivery method with each subscription configuration.

Subscription Maintenance

The system is required to manage subscriptions as follows:

21. Users shall be able to perform subscription maintenance tasks through email messages or through the Web interface.
22. The system shall have the ability to authenticate subscription maintenance actions.
23. Users shall have the ability to view their subscription configurations.
24. Users shall have the ability to change their subscription configurations.
25. Users shall have the ability to deactivate their subscription configurations.

Subscription Distribution

The system is required to deliver subscription data to the user as follows:

26. Subscription data shall be delivered through standard UNIX mail to the email address associated with the subscription.

27. Subscription data may be delivered via FTP if the data-set to be returned is deemed too large to be sent by email or if the user identifies FTP as the preferred delivery method. Local drop directories and FTP push shall be supported as FTP delivery options.
28. The system shall provide an “alert” mechanism, which will notify users of important system changes.
29. The system shall have a mechanism for the request and delivery of missed messages.
30. The system shall have a unique, consecutive identification number per user/subscription.
31. The system shall recover and distribute products interrupted by a failure of hardware or software at the IDC.
32. The inclusion of customizable, product-specific headers shall be supported. At a minimum, headers shall display the date and time at which the product was prepared for delivery.
33. If a subscription request returns no data, a “blank” message indicating that no data exists shall be sent.
34. The user shall have the ability to disable the feature described by requirement 33..
35. The subscription should be able to support multiple formats for waveform products.
36. A subscription shall return data for any/all standard IDC products.

Logging/Monitoring

This system is required to log and monitor subscription requests as follows:

37. The system shall log relevant information about subscriptions processed including query data, recipient, time/date, and size.
38. The system shall log all configuration/maintenance actions.
39. Adequate security precautions shall be taken to ensure the privacy of the log.

Requirements ▼

- 40. Logs shall be kept for an operator-definable period of time.
- 41. Users shall have access to all information in the log regarding their configuration, maintenance, and distribution profiles.
- 42. The system shall allow for real-time system monitoring by authorized individuals.

SYSTEM REQUIREMENTS

The Subscription Subsystem will meet the following system requirements:

- 43. The system shall use the IDC ORACLE database for all permanent data requiring transaction management.
- 44. The system shall use the Generic Database Library for database operations.
- 45. The system shall use command line arguments to pass run-time data to the application software. These arguments will be provided in “par” files and standard IDC software will be used for reading and parsing these files.
- 46. The system shall be compatible with the GSE 2.0 message structures.
- 47. The system shall use the Message Subsystem to provide subscription requests.
- 48. The Message Subsystem will format and deliver (by UNIX mail) filled subscription orders to subscribers. Orders that are too large to be handled by email will be placed in files in a location known to subscribers. The subscribers will be notified when the files are available for FTP delivery.

REQUIREMENTS TRACEABILITY

The following tables trace the requirements of the Subscription Subsystem to components and describe how the requirements are fulfilled.

TABLE 15: TRACEABILITY OF GENERAL REQUIREMENTS

	Requirement	How Fulfilled
1.	The system shall require minimum operator intervention.	<i>ParseSubs</i> will process the subscriptions without operator intervention (once authentication is resolved).
2.	Impact upon other IDC resources (such as the Database Management System) shall be minimized.	System resources are minimized by processing on a product level so that an individual product can be received by multiple recipients, thus minimizing database activity and disk usage if the products are identical.
3.	The system shall be extensible.	The table structure allows for additional functionality to be added without redesigning the tables.
4.	The system shall make maximum use of other elements of the Data Services CSCI.	The system uses <i>AutoDRM</i> as the database request module, which is the one-time requestor for timeseries processing.
5.	The system shall be scalable.	The slower portions of the system, namely the processing of the products, will handle the product types independently, possibly on different machines. However, an additional database connection will be required with each independent process.

Requirements ▼

TABLE 16: TRACEABILITY OF FUNCTIONAL REQUIREMENTS: STORAGE OF USER PROFILES AND SUBSCRIPTION RECORDS

	Requirement	How Fulfilled
6.	Users shall be identified by an email address.	The subs table contains the email address.
7.	The system shall associate additional point-of-contact information with each user.	This requirement is not satisfied in this release.
8.	The system shall have the ability to associate subscription records with user profiles.	The <i>userid</i> connects subs with subs-user .
9.	The system shall have the ability to authenticate and log a user profile insertion or update.	This requirement is fulfilled through authentication in the Message Sub-system. The method of authentication and requirements for authentication will be addressed independently of the Subscription Sub-system.
10.	Users shall have the ability to configure an indefinite number of subscriptions.	The subs table is open ended.
11.	The system shall maintain the date that a subscription was originally configured.	The <i>ondate</i> field in the subs table contains this information.
12.	The system shall maintain the date that a subscription configuration was last changed.	The <i>lddate</i> field in the subs table contains this information.
13.	The system shall have the ability to determine whether or not a subscription is active.	The <i>status</i> field in the subs table contains this information.

**TABLE 17: TRACEABILITY OF FUNCTIONAL REQUIREMENTS:
SUBSCRIPTION CONFIGURATION**

	Requirement	How Fulfilled
14.	Users shall be able to configure subscriptions through the Web interface.	A form will be created on a Web page and will be sent initially via email to the Message Subsystem. A Web interface will be developed later for the Message Subsystem.
15.	Users shall be able to configure subscriptions through email.	The Message Subsystem is the front end and is based on email.
16.	The system shall have the ability to authenticate a subscription configuration.	Authentication will be added to the Message Subsystem.
17.	The subscription may be configured to return data regularly. Regularity is defined as the ability to send data as soon as it is received, daily, weekly, or monthly.	The <i>cycle_time</i> field in the product table contains this information.
18.	Provisions shall be made to distribute data based upon reaching a defined size limit.	This requirement is not satisfied in this release.
19.	Provisions shall be made to deliver data, in conjunction with requirements 17. and 18., on a given day, date, and/or time as they are requested for some products.	The column <i>timenextsend</i> in the productcriteria table tracks when the system should next distribute a product. When that time is less than the current time, the product will be processed.
20.	Users may specify a destination/delivery method with each subscription configuration.	The <i>deliv_meth</i> column in the subs table will track the destination/delivery method; the Message Subsystem will deliver the product.

**TABLE 18: TRACEABILITY OF FUNCTIONAL REQUIREMENTS:
SUBSCRIPTION MAINTENANCE**

	Requirement	How Fulfilled
21.	Users shall be able to perform subscription maintenance tasks through email messages or through the Web interface.	(same as 14. and 15.)
22.	The system shall have the ability to authenticate subscription maintenance actions.	(same as 16.)
23.	Users shall have the ability to view their subscription configurations.	Viewing will be an option in <i>Parse-Subs</i> (See "DataReady_to_Prodtrack" on page 30.).
24.	Users shall have the ability to change their subscription configurations.	Change will be an option in <i>Parse-Subs</i> (See "DataReady_to_Prodtrack" on page 30.).
25.	Users shall have the ability to deactivate their subscription configurations.	The <i>status</i> field in the subs table fills this requirement.

**TABLE 19: TRACEABILITY OF FUNCTIONAL REQUIREMENTS:
SUBSCRIPTION DISTRIBUTION**

	Requirement	How Fulfilled
26.	Subscription data shall be delivered through standard UNIX mail to the email address associated with the subscription.	The Message Subsystem delivers all mail (through <i>MessageSend</i>).
27.	Subscription data may be delivered via FTP if the data-set to be returned is deemed too large to be sent by email or if the user identifies FTP as the preferred delivery method. Local drop directories and FTP push shall be supported as FTP delivery options.	<i>MessageSend</i> formats the FTP Log. <i>FTPPush</i> will be added to <i>MessageSend</i> .
28.	The system shall provide an “alert” mechanism, which will notify users of important system changes.	This mechanism will be a type of subscription. The protocol is to be defined.
29.	The system shall have a mechanism for the request and delivery of missed messages.	The Message Subsystem stores the message on disk. One option under <i>ParseSubs</i> will be to resend a message. When recipients receive a subscription that skips <i>delivid</i> , they request redelivery of that message.
30.	The system shall have a unique, consecutive identification number per user/subscription. ¹	The <i>outgoing_subject</i> field of the message will contain this information, which will also be available in the <i>delivid</i> field of the productcriteria table.
31.	The system shall recover and distribute products interrupted by a failure of hardware or software at the IDC.	The <i>status</i> field in the prodtrack table will indicate if the data have been processed or not. <i>SubsWatch</i> will verify that all entries have been processed.
32.	The inclusion of customizable, product-specific headers shall be supported. At a minimum, headers shall display the date and time at which the product was prepared for delivery.	<i>AutoDRM</i> parameters will be printed in the outgoing message (See “SubsProcess” on page 33.).

Requirements ▼

TABLE 19: TRACEABILITY OF FUNCTIONAL REQUIREMENTS:
SUBSCRIPTION DISTRIBUTION (CONTINUED)

	Requirement	How Fulfilled
33.	If a subscription request returns no data, a "blank" message indicating that no data exists shall be sent.	<i>AutoDRM</i> will fulfill this requirement (See "SubsProcess" on page 33.).
34.	The user shall have the ability to disable the feature described by requirement 33..	The <i>suppress_empty</i> column in the product table disables the sending of empty messages. <i>AutoDRM</i> parameters will control the mechanics (See "SubsProcess" on page 33.).
35.	The subscription should be able to support multiple formats for waveform products.	<i>AutoDRM</i> parameters will control the mechanics (See "SubsProcess" on page 33.).
36.	A subscription shall return data for any/all standard IDC products.	Either database triggers or hooks into the database will fulfill this requirement.

1. The original requirement was per user/subscription. That requirement resulted in an unacceptably large tracking table. This change was approved at the design review.

**TABLE 20: TRACEABILITY OF FUNCTIONAL REQUIREMENTS:
LOGGING/MONITORING**

	Requirement	How Fulfilled
37.	The system shall log relevant information about subscriptions processed including query data, recipient, time/date, and size.	Message Subsystem tracking coupled with the subtrack table will fulfill this requirement.
38.	The system shall log all configuration/maintenance actions.	Log files will fulfill this requirement.
39.	Adequate security precautions shall be taken to ensure the privacy of the log.	Operating system and database security will fulfill this requirement.
40.	Logs shall be kept for an operator-definable period of time.	
41.	Users shall have access to all information in the log regarding their configuration, maintenance, and distribution profiles.	<i>ParseSubs</i> will have the option to access this information. <i>ParseSubs</i> will return all of the information in the database regarding the subscription.
42.	The system shall allow for real-time system monitoring by authorized individuals.	A monitoring tool will be developed to fulfill this requirement (See "SubsProcess" on page 33.).

References

The following sources are referenced in this document or supplement it:

- [And90a] Anderson, J., Farrell, W.E., Garcia, K., Given, J., and Swanger, H., *CSS Version 3 Database: Schema Reference Manual*, Science Applications International Corporation, 1990.
- [And94] Anderson, J., Mortel, M., MacRitchie, B., and Turner, H., *Generic Database Interface (GDI) User Manual*, Science Applications International Corporation, SAIC-93/1001, 1994.
- [Car97] Carter, J., and Bowman, J. R., *IDC Database Schema*, CMR-97/28, 1997.
- [DOD94a] Department of Defense, "Software Design Description," *Military Standard Software Development and Documentation*, MIL-STD-498, 1994.
- [DOD94b] Department of Defense, "Software Requirements Specification," *Military Standard Software Development and Documentation*, MIL-STD-498, 1994.
- [GSE95b] Group of Scientific Experts, *GSETT 3 Documentation, Volume Two: Operations*, CRP/243, 1995.
- [IDC3.4.1] Science Applications International Corporation, Pacific-Sierra Research Corporation, *Formats and Protocols for Messages*, SAIC-98/3004, PSR-98/TN1129, 1998.
- [IDC3.4.2] Science Applications International Corporation, *Formats and Protocols for Continuous Data*, SAIC-98/3005, 1998.

References ▼

- [IDC5.1.1] Science Applications International Corporation, Pacific-Sierra Research Corporation, *Database Schema (Part 1 and Part 2)*, SAIC-98/3009, PSR-98/TN1127, 1998.
- [Sal97] Salzberg, D., "Adding Station Capability and Flat Products to the Subscription Subsystem," *Configuration Control Board Memorandum*, CCB-PRO-97/9, Centre for Monitoring Research, 1997.

Glossary

A

arrival

A signal that has been associated to an event. In the first instance, this is performed automatically by the Global Association (GA) software. Later in the processing, many arrivals are confirmed and improved by visual inspection.

B

bulletin

Chronological listing of event origins spanning an interval of time. Often, the specification of each origin, or event, is accompanied by the event's arrivals, and sometimes with the event's waveforms.

C

child process

UNIX process created by the *fork* routine. The child process is a snapshot of the parent at the time it called *fork*.

CMR

Center for Monitoring Research.

Configuration Control Board

Organizational body that approves and releases new versions of software.

COTS

Commercial-Off-the-Shelf; terminology that designates products such as hardware or software that can be acquired from existing inventory and used without modification.

CSC

Computer Software Component.

CSCI

Computer Software Configuration Item.

CSP

Conference of States Parties; the principal body of the CTBTO consisting of one representative from each State Party accompanied by alternate representatives and advisers. The CSP is responsible for implementing, executing, and verifying compliance with the Treaty.

CTBT

Comprehensive Nuclear Test-Ban Treaty (the Treaty).

CTBTO

Comprehensive Nuclear Test-Ban Treaty Organization; Treaty User group that consists of the Conference of States Parties (CSP), the Executive Council, and the Technical Secretariat.

Glossary ▼

D**detection**

Probable signal that has been automatically detected by the Detection and Feature Extraction (DFX) software.

download

To copy a document or application from one computer to another.

E**email**

Electronic mail.

F**FTP**

File Transfer Protocol; a method for transferring files between computers.

G**GB**

Gigabyte.

GDI

Generic Database Interface.

GSE

Group of Scientific Experts.

GSETT-3

Group of Scientific Experts Third Technical Test.

H**HTML**

Hypertext Markup Language; formatting language of the Web.

hydroacoustic

Pertaining to sound in the ocean.

I**IDC**

International Data Centre.

IDC Operators

Technical staff that install, operate, and maintain the IDC systems and provide additional technical services to the individual States Parties.

IMS

International Monitoring System.

IMS Operators

Technical staff that operate and monitor the IMS facilities.

infrasonic

Pertaining to low frequency (sub-audible) sound in the atmosphere.

infrastructure

Foundation and essential elements of a system or plan of operation.

Internet

World-wide network of computers linked by means of the IP protocol.

M**mb**

Magnitude of an event based on measurements of seismic bodywaves.

MB

Megabyte.

ML

Magnitude of an event based on waves measured near the source.

monitoring system

See IMS and RMS.

Ms

Magnitude of an event based on measurements of seismic surface waves.

N**NDC**

National Data Center.

O**online**

Logged onto the network or having unspecified access to the Internet.

Operations Manuals

Treaty-specified, formal documents that describe how to provide data, receive IDC products, access the IDC database, and evaluate the performance of the IDC.

origin

Place and time of a seismic, hydroacoustic, or infrasonic event.

OSI

On-site Inspection.

P**parameter (par) file**

ASCII file containing values for parameters of a program. Par files are used to replace command line arguments. The files are formatted as a list of [token = value] strings.

PIDC

Prototype International Data Centre.

PIDC System Developers

Contractors and other organizations who are developing and testing components of the PIDC technology.

Preparatory Commission

Preparatory Commission for the CTBTO; new international body funded by States Parties to prepare for implementation of the Treaty. This body will become the CTBTO after entry-into-force of the Treaty.

R**REB**

Reviewed Event Bulletin; the bulletin formed of all events that have passed analyst inspection and quality assurance review. The REB runs 48 hours behind real time.

Glossary ▼

S

SEL1

Standard Event List 1; the bulletin created by total automatic analysis of continuous timeseries data. Typically, the list runs one hour behind real time.

SEL2

Standard Event List 2; the bulletin created by totally automatic analysis of both continuous data and segments of data specifically down-loaded from stations of the auxiliary seismic network. Typically, the list runs four hours behind real time.

SEL3

Standard Event List 3; the future name for the Reviewed Event Bulletin.

SMTP

Simple Mail Transfer Protocol.

States Parties

Nations which have signed and ratified the Comprehensive Nuclear Test-Ban Treaty.

T

taxonomy

Systematic arrangement; classification.

Treaty

Comprehensive Nuclear Test-Ban Treaty (CTBT).

Treaty Users

CTBTO and States Parties.

U

UN/CD

United Nations Conference on Disarmament.

W

Web

World Wide Web; a graphics-intensive environment running on top of the Internet.

workstation

High-end, powerful desktop computer preferred for graphics and usually networked.